**Experiment 1:** WAP to implement the following scenarios. Take all the input from user, nothing should be imagined or hard coded.

1. Transpose of a matrix
2. Check if a matrix is Syymetrical or not
3. Check the inverse of a matrix

**Solution:-**

**Code:-**

#include <stdio.h>

int main() {

    int a[2][2];

    int trans[2][2]; // Declare 'trans' after initializing 'i' and 'j'

    int i, j,symmetric=1;

    double determinant,inverse[2][2];

    printf("Enter matrix elements:\n");

    for (i = 0; i < 2; i++) {

        for (j = 0; j < 2; j++) {

            scanf("%d", &a[i][j]);

        }

    }

    printf("Matrix elements:\n");

    for (i = 0; i < 2; i++) {

        for (j = 0; j < 2; j++) {

            printf("%d\t", a[i][j]);

        }

        printf("\n");

    }

    for (i = 0; i < 2; i++) {

        for (j = 0; j < 2; j++) {

            trans[j][i] = a[i][j];

        }

    }

    printf("Transpose elements:\n");

    for (i = 0; i < 2; i++) {

        for (j = 0; j < 2; j++) {

            printf("%d\t", trans[i][j]);

        }

        printf("\n");

    }

        for(i=0;i<2;i++)

        {

            for(j=0;j<2;j++)

            {

                if(a[i][j]!=trans[i][j])

                {

                    symmetric=0;

                    break;

                    }

            }

            if(!symmetric)

            {

                printf("the matrix ix symmetric\n");

            }

            else{

                printf("the matrix is not symmetric\n");

            }

            }

            printf("enter matrix element\n");

            for (i = 0; i < 2; i++) {

        for (j = 0; j < 2; j++) {

            scanf("%d", &a[i][j]);

        }

    }

    // Calculate the determinant

    determinant = a[0][0] \* a[1][1] - a[0][1] \* a[1][0];

    if (determinant == 0.0) {

        printf("Matrix is singular, inverse doesn't exist.\n");

    } else {

        // Calculate the inverse

        inverse[0][0] = a[1][1] / determinant;

        inverse[0][1] = -a[0][1] / determinant;

        inverse[1][0] = -a[1][0] / determinant;

        inverse[1][1] = a[0][0] / determinant;

        // Display the inverse matrix

        printf("Inverse matrix:\n");

        for (i = 0; i < 2; i++) {

            for (j = 0; j < 2; j++) {

                printf("%lf\t", inverse[i][j]);

            }

            printf("\n");

        }

    }

    return 0;

}

}

**Experiment 2:-** WAP to merge two arrays and append them in the following order.

1. Add the first array to the end of another one
2. Add Second Array to the end of the first one
3. Merge the arrays and sort them.

**Solution:-**

**Code:-**

**Experiment 3:-**. WAP using pointers to find the smallest number in an array using pointer.

**Soultion:-**

**Code:-**

#include <stdio.h>

int main() {

    int n;

    printf("Enter the number of elements in the array: ");

    scanf("%d", &n);

    int arr[n];

    printf("Enter the array elements:\n");

    for (int i = 0; i < n; i++) {

        scanf("%d", &arr[i]);

    }

    int \*ptr = arr;

    int smallest = \*ptr;

    for (int i = 1; i < n; i++) {

        if (\*(ptr + i) < smallest) {

            smallest = \*(ptr + i);

        }

    }

    printf("The smallest number in the array is: %d\n", smallest);

    return 0;

}